DSM におけるドメインフレームワークについて

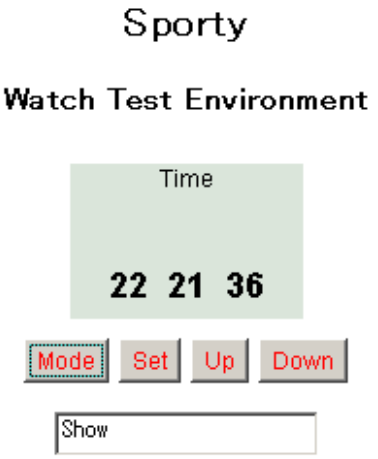
ドメインスペシフィックなモデル（DSM）からコードを生成させるコツは、アプリケーション内から共通した処理や関数を抽出してドメインフレームワークの部品にすることです。そうすることで、モデルから生成させるコードは、部品を呼び出す為の処理や条件判断文のみで済むようになり、ジェネレータの設定（スクリプト）は、以下のように簡単なものになります。

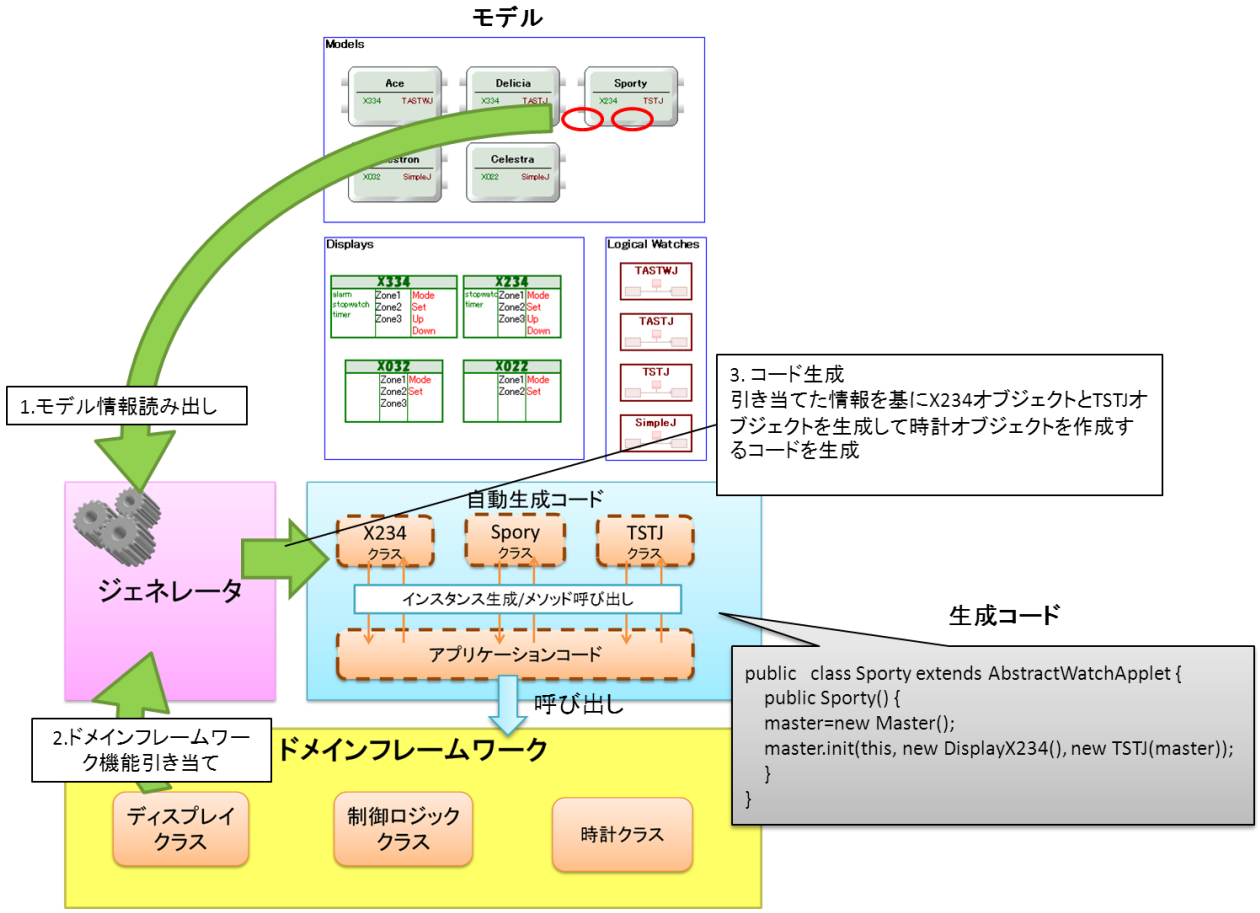
- 共通の処理、関数をメタモデルに定義されたルールや条件などに従って呼び出す
- 呼び出す時のルールや条件はパラメータとして共通の処理、関数に引き渡す
- モデル内で定義されるモデル部品のパラメータも、そのまま共通の処理、関数に引き渡す
- モデルの意図に沿ってコードとして構成する

結果、アプリケーション内の似通ったコード領域は、モデル部品とそのパラメータの組合せで包括されます。そのため細かいモデル部品であふれてしまうことは有りません。コードの実装と比較して抽象度が上がり、作業効率は飛躍的に向上します。同じコードを繰り返し記述する単純作業からも解放されます。同時に、モデリング環境にルールやコツが包括されているため、コーディングミスや単純なバグ、ライブラリやコンポーネントの誤用を回避する効果も得られます。

以下インストール内の Watch Example（デジタル時計）で、モデルから生成させるコード（部品を呼び出す為の処理や、条件判断文）と、ドメインフレームワークの役割について簡単に説明します。
（注：コード生成動作に関しては、別途、“ジェネレータの動作について”をご参考下さい）

Watch Example は、“Watch モデルから生成されるコード”、“Java VM/ Java フレームワーク”、“ドメインフレームワーク”の 3 層構造で、デジタル時計を Java VM 上で実現します。



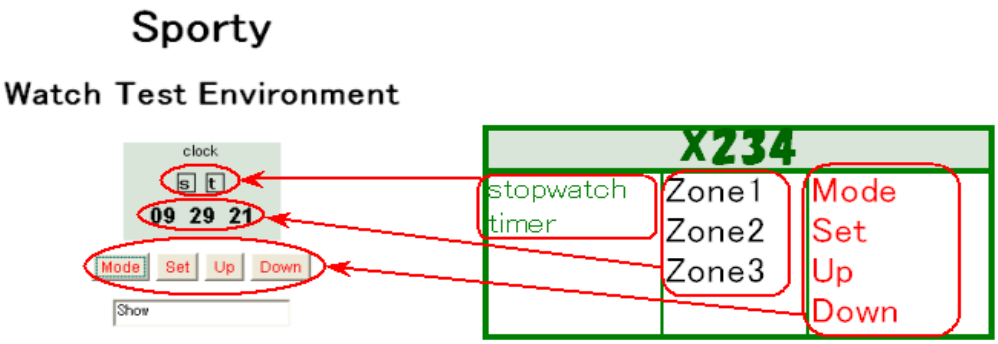


上図 **Sporty**（スポーツ時計）モデルには、部品としてディスプレイモデル **X234** と、時計機能モデル **TSTJ** を組み込んでいます。この **Sporty** から生成されるコードは **X234** と **TSTJ** に対するオブジェクト **DisplayX234()** と **TSTJ(master)** の生成を new で行うだけのものです。

ドメインフレームワークに部品が用意されていることにより、ジェネレータが行う処理はモデルの情報から必要なドメインフレームワークの部品を選び、それらの部品を組み合わせるだけのシンプルなものとなります。

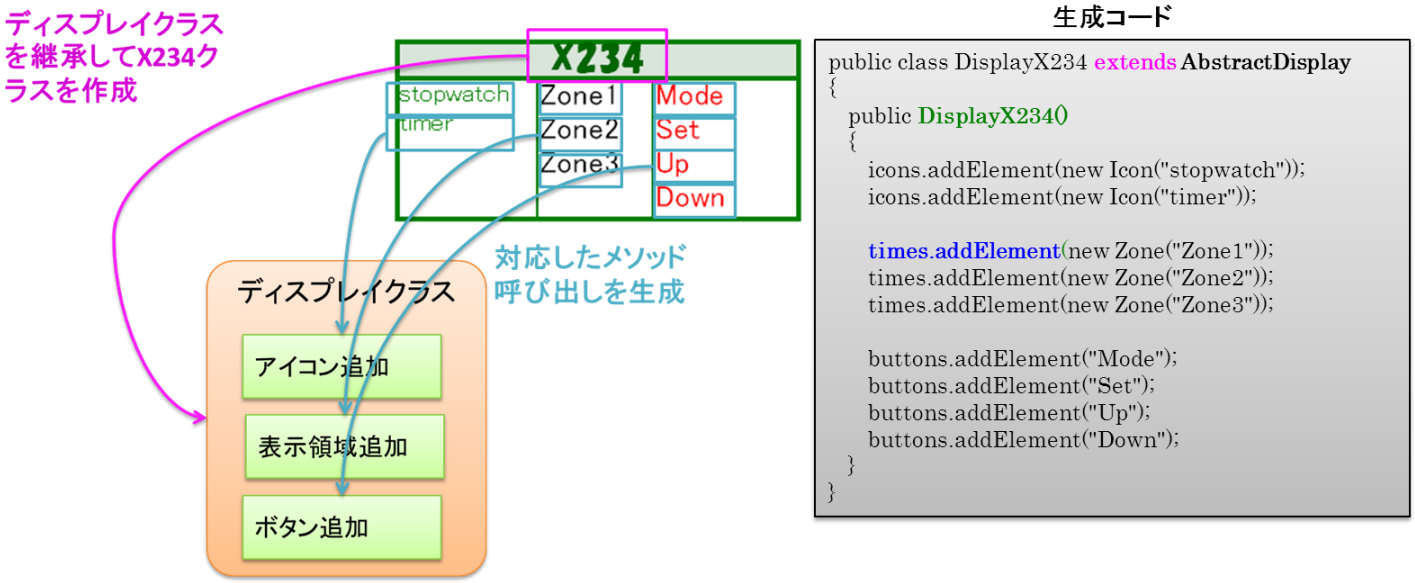
```
public    class Sporty extends AbstractWatchApplet {
    public Sporty() {
        master=new Master();
        master.init(this, new DisplayX234(), new TSTJ(master));
    }
}
```

さらにディスプレイモデルのコード生成の詳細を説明します。X234 は、時計の画面を定義するモデルです。これを実現する為に、“表示領域確保”、“データの参照”、などの処理のコードをドメインフレームワーク部品として用意しています。



ディスプレイモデルX234 上の Zone 1,2,3 は、時、分、秒のための表示領域です。これに対し、モデルから生成される以下 DisplayX234() の中身は、表示領域を生成するドメインフレームワーク機能の呼出しとなります。

Zone1 から生成されるコード times.addElement(new Zone("Zone1")); から
times.addElement(new Zone("Zone1"));
（**times** は、ドメイン固有に書いたコード ⇒ Zone の表示領域確保するためのオブジェクト）
（**addElement** は、Java の標準関数 ⇒ Zone オブジェクトの追加）
の処理を生成し Zone1 の表示領域が確保します。



ここで times は Zone の表示領域を確保するための可変長配列変数です。ソースコードの実装では、このような配列変数の確保を行う処理はプログラマごとで個々にコーディングされたりしますが、この DSM の例では、基本クラス AbstractDisplay で配列変数の確保を実装し、基本クラスを継承した DisplayX234 クラスを作成することで配列変数の確保を実現しています。

```
abstract class AbstractDisplay
{
        Vector icons     = new Vector();
        Vector times     = new Vector();
        Vector buttons = new Vector();
}
```

times 可変長配列の確保

このように AbstractDisplay 基本クラスをドメインフレームワークの部品として登録し、モデルから生成されるコードに組み込まれるようにジェネレータを設定すれば、アプリケーションごとで個々に記述されるような処理も共通部品化することができます。結果としてコードジェネレータの定義は容易になり、また共通部品として再利用も可能となります（更なるソフトウエアプロダクトライン開発）。

最後に参考として、この例におけるディスプレイへの表示方法を説明します。確保された表示領域をディスプレイに表示する方法は、ターゲットプラットフォームによって様々でしょう。この例では表示領域が確保されると、その領域が一定時間ごとに JavaVM のプラットフォームによって更新されます。表示領域の更新が発生すると、フレームワーク内の以下のコードが呼ばれ、DisplayX234 オブジェクトの times を参照しながら、JavaVM 上の表示を更新します。times の各 Zone の内容を参照し、その値を JavaVM 特定の領域に描画しています。

```
for (Enumeration e = master.display.times.elements() ; e.hasMoreElements() ;)
{
    String zoneString;

    Zone zone=(Zone)(e.nextElement());
    if (zone.blinking && ((System.currentTimeMillis())%1000)<500)
    {
        zoneString="   ";
    }  else  {
        zoneString=zone.zoneValue;
    }

    g.drawString(zoneString, x, y+jdk1);
    x+=3*charWidth;
}
```

DisplayX234 オブジェクトへの参照

参照した値を JavaVM で描画する

富士設備工業株式会社 電子機器事業部
〒591-8025 大阪府堺市北区長曽根町1928-1
Tel: 072-252-2128 www.fuji-setsu.co.jp